REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate to any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	05 June 2002	Phase I Option Addendum 2002 FEB 25 - MAY 25

4. TITLE AND SUBTITLE
Compressed Internet Protocol (IP) Data via Geosynchronous Earth Orbit (GEO) Satellite Circuits

5. FUNDING NUMBERS
(C) Contract Number
N00039-01-C-3208

(PR) Project Number
N00039-01-PR-D61106

6. AUTHOR(S)
Mr. Youngki Hwang, Mr. Stuart W. Card, Mr. Fred W. Tims

7. PEFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Critical Technologies Inc.
Suite 400, Technology Center
4th Floor, 1001 Broad Street
Utica, NY 13501

8. PERFORMING ORGANIZATION REPORT NUMBER
ATC-P1-OA

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Space and Naval Warfare Systems Command
4301 Pacific Highway
San Diego, CA 92110-3127

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
This Option Addendum applies to the Final Technical Report of the subject contract, and describes the work performed under the Phase I Option exercised by the Government subsequent to the submission of the FTR.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Proposed availability statement, subject to Government approval -- Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

To enable Internet Protocol applications in the submarine environment, Critical Technologies Inc. (CTI) has defined requirements for an integrated system of enhanced Data Link, Network and Transport protocols (in OSI Layers 2-4) and an innovative Application Traffic Controller (ATC, in OSI Layers 5-7). It integrates caching, compression, prioritization and queuing to optimize link utilization. The Compression Controller segment will provide users with compression benefits without requiring them to manually invoke compression utilities. It will automatically and transparently, based upon policy rules, for each data object: determine whether the data requires compression, and if so, which algorithms and parameters are most appropriate for that data; and then dispatch the object to the most appropriate available external compression utility. Navy system administrators will define the rules, based on Application Layer data object type, size, format, content, intended use, importance, source/destination, etc., per approved doctrine. In addition to automating compression, it will provide better control over the tradeoffs among delivered data quality, timeliness and wireless resource consumption. In the Phase I Option, CTI developed a Compression Controller pre-prototype and successfully tested it over UHF DAMA MILSATCOM.

14. SUBJECT TERMS
submarine communications, SATCOM, Internet, protocol, TCP/IP, compression, policy

15. NUMBER OF PAGES
18 inc. title, plus this SF298

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18

20020610 047

Navy Small Business Innovation Research (SBIR) Program

Compressed Internet Protocol Data Via Geosynchronous Earth Orbit Satellite Circuits

SPAWAR SBIR N01-052 Phase I Option Addendum to the Final Technical Report

CDRL Item Number: A005

2002-Jun-05

Sponsored by:
Space and Naval Warfare Systems Command
Attn: Ray Potts, PMW 173-3
4301 Pacific Highway
San Diego, CA 92110-3127

Contract Number: N00039-01-C-3208
Contract Option Dollar Amount $29,883
Competitive Award

Prepared by:
Youngki.Hwang@critical.com, Stu.Card@critical.com, Fred.Tims@critical.com

Critical Technologies Inc.
Suite 400 Technology Center
4th Floor 1001 Broad Street
Utica, NY 13501
315-793-0248 / FAX –9710
http://www.critical.com

Proposed availability statement, subject to Government approval:
Approved for public release; distribution unlimited.
Security Markings or Handling Restrictions: N/A

Table of Contents

1 Introduction

This document presents the design of the pre-prototype Compression Controller proposed as an optional task for Phase I of the "Compressed Internet Protocol Data Via Geosynchronous Earth Orbit Satellite Circuits" SBIR project. The Compression Controller is a segment of the Application Traffic Controller (ATC) designed during Phase I and depicted in Appendix A.

The pre-prototype Compression Controller was implemented and debugged in the CTI laboratory under Linux 2.4 in conjunction with the Apache Web Server (v2.0.32). It was then installed in the Network Design Facility (NDF) at the Rome Research Site (RRS) of Air Force Research Laboratory (AFRL) and successfully tested over their satellite resources (see 6).

This document and associated pre-prototype software and accompanying installation/use notes is presented in satisfaction of CLIN 005 of Contract Number: N00039-01-C-3208.

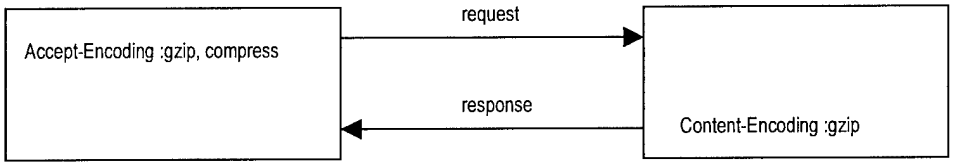
2 Background

The objective of the Compression Controller segment of the ATC is to minimize traffic loads over disadvantaged communications links by optimally compressing files before transmission and decrompressing after transmission. An associated design goal is to automatically select and perform the optimal compression algorithm in a manner transparent to the user. Further discussion of the Compression Controller can be found in Appendix B.

3 Current HTTP/1.1 Compression Mechanisms.

All modern browsers which are HTTP/1.1 compliant, released since early 1999, are capable of receiving compressed internet content. The HTTP/1.1 Standard (RFC 2068) defines procedures to be followed to compress HTTP content being transferred from web server down to browsers using public domain compression algorithms (GZIP compression libraries such as ZLIB). There are two compression mechanisms defined in HTTP/1.1, ***Content-Encoding*** and ***Transfer-Encoding.*** The HTTP 1.1 message format is described in Appendix C.

3.1 Content-Encoding



"Content-Encoding" is an IETF convention for sending pre-compressed content over the Internet. The files being sent have already been cached in their compressed format and are generally static. Following this convention, a client browser that is capable of handling compressed files notifies the server of its capabilities by supplying a header field in the request for the file, identifying which compression methods it can handle, for instance:

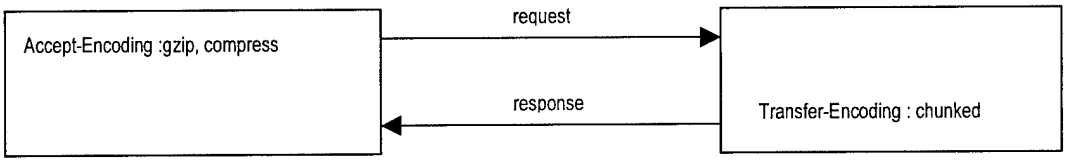
Accept-Encoding : gzip, compress (The arguments identify which compression methods it supports; in this case gzip and compress)

The server will then look in its cache for a pre-compressed version of the requested file (in this example, it would look for either a "gzip" or a "compress" file). If it finds one it will send it rather than the much larger uncompressed version.

> **Note:** RFC 2068 actually defines this construct as restrictive, rather than enabling, i.e., in section 14.3... "If no Accept-Encoding header is present in a request, the server MAY assume that the client will accept any content encoding. " If an Accept-Encoding header is present, but no compression types are specified in the field value, none are acceptable and the file must be sent uncompressed. The paragraph also states that if a compression type (or types) *is* specified, and no version of the file in a specified type is present in the server cache, an error message should be sent to the client browser. This seems counter-intuitive and conflicts with our discussion above. It would seem to make better sense, as we have assumed above, that, if the cache contains no version of the file in a specified compression format, the uncompressed version should be sent rather than an error message. These, and other issues, such as the degree of conformance to the specification of operating systems (Linux, Windows), Web servers (Apache, MS-IIS) and Web browsers (Netscape, Explorer), will be resolved early in Phase II of the project.

Although still supported, Content-Encoding is of minor importance, generally, due to the fact that most pages served up today on the Web are dynamically generated, thus not candidates for pre-compression. It does, however, have applicability to our Compression Controller design, as will be discussed later.

3.2 Transfer-Encoding



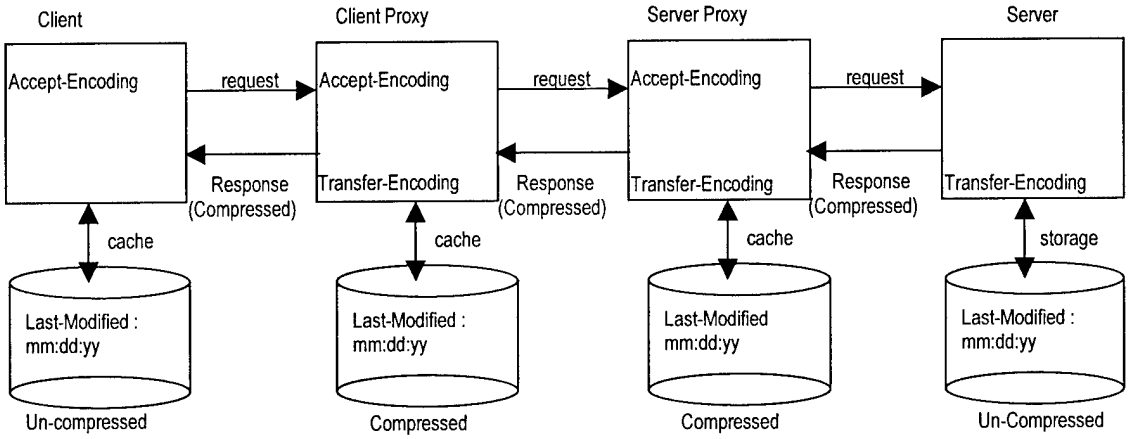
"Transfer-Encoding" is a method whereby files are compressed just before being sent to the client. This method addresses the needs of tailor-made pages created dynamically, which are not amenable to pre-compression. Those Web servers that support Transfer-Encoding use a compression approach called "chunking", where the file is first broken into "chunks" the size of a transmission unit, each chunk being compressed just before transmission. Chunking is the only compression mechanism offered for Transfer-Encoding, probably because, although it is not efficient from a compression ratio standpoint, it is the best that can be done in the limited time available to busy Web servers, which serve a multitude of users in real time.

4 Some Possible Approaches

Based on current conventions, we can consider several approaches to provide HTTP compression mechanisms. There are many permutations, but the following are considered sufficient to illustrate the design decision.

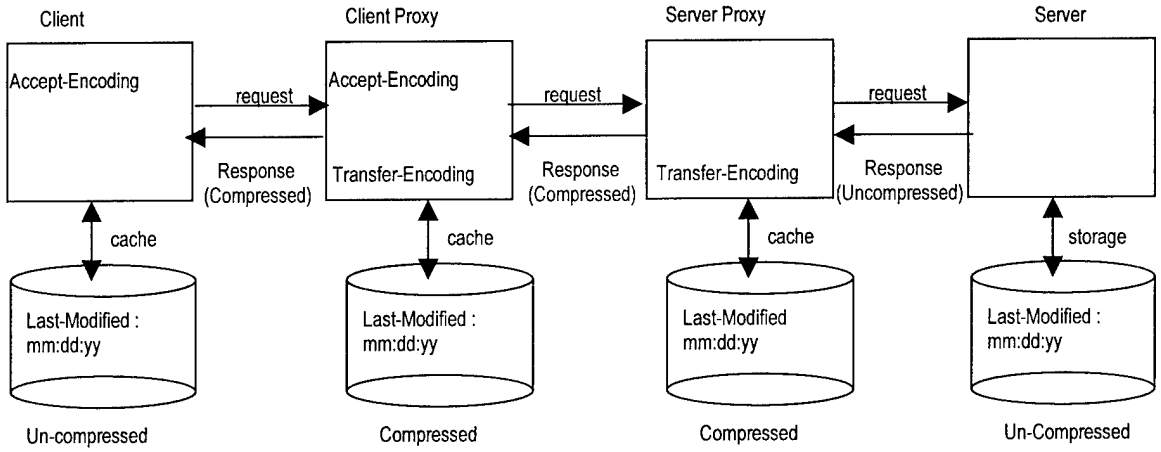
4.1 End-to-End HTTP Compression with No proxy operations

Assuming a server capable of Transfer-Encoding, this approach is most commonly used in today's Internet.



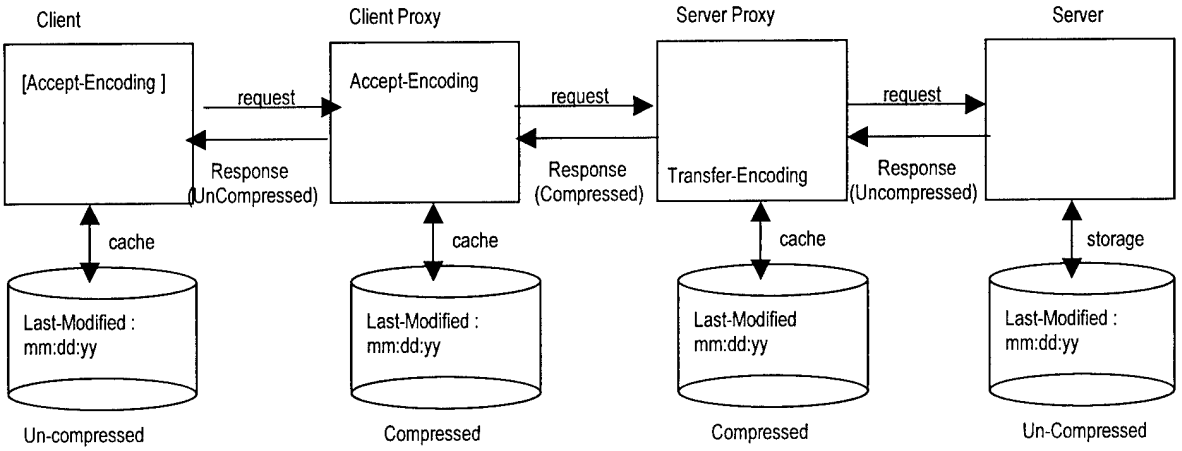
The server compresses the file and sends the compressed version to the client. There are no proxy compression operations. The client's web browser must have the ability to decompress. Files are cached in compressed format in the proxies to avoid a compression operation when the request is satisfied from either the near or the far proxy cache. Only "chunking" compression is supported.

4.2 Proxy-to-End HTTP compression



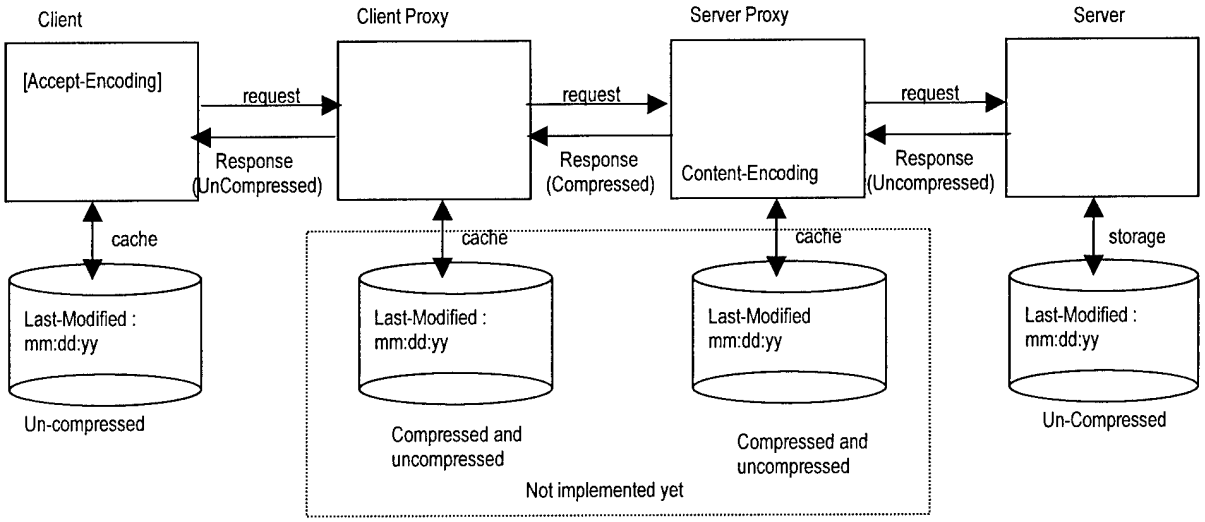
The server proxy compresses the file and sends the compressed version to the client proxy, which then sends it to the client. No client proxy compression operations are involved. The client's web browser must have the ability to decompress. As in 4.1, files are cached in compressed format in the proxies to avoid a compression operation when the request is satisfied from either the near or the far proxy cache. Only "chunking" compression is supported.

4.3 *Proxy-to-Proxy HTTP Compression*



Compression is performed between proxies only. The server proxy compresses and the client proxy decompresses. *No operations in the endpoints are involved.* Files are cached in compressed format in the proxies to avoid a compression operation when the request is satisfied from the far proxy cache. In this approach, the native capability of the Server Proxy to perform Transfer-Encoding is employed, thus only "chunking" is available.

5 Innovative Approach



5.1 Operation

In this approach, a refinement of that described in 4.3, the Content-Encoding model is employed, which gives us an immediate choice of *gzip* or *compress*, and future expansion capabilities[1]. There is a flavor of Transfer-Encoding in this approach, however, in that, if the file does not already exist in a suitably compressed format in the server cache, it will be compressed and stored there before transmission and for later use for subsequent requests. Unlike a general-purpose Web server, which must serve many clients in real time, the Compression Controller component of the ATC has a relatively limited number of users and can afford the time to do a proper job of compressing files when it must.

In all cases, whether or not the client or the server has the ability to decompress or to compress, if the request can be satisfied from neither cache:

When the server proxy receives a REQUEST packet from the client proxy:

- It changes the header field of the REQUEST packet, i.e. it sets the "accept-encoding" value as null, so that the server will be inhibited from invoking any compression operations,
- It forwards the changed REQUEST packet to the server.

When the server proxy receives a RESPONSE packet with uncompressed original data from the server:

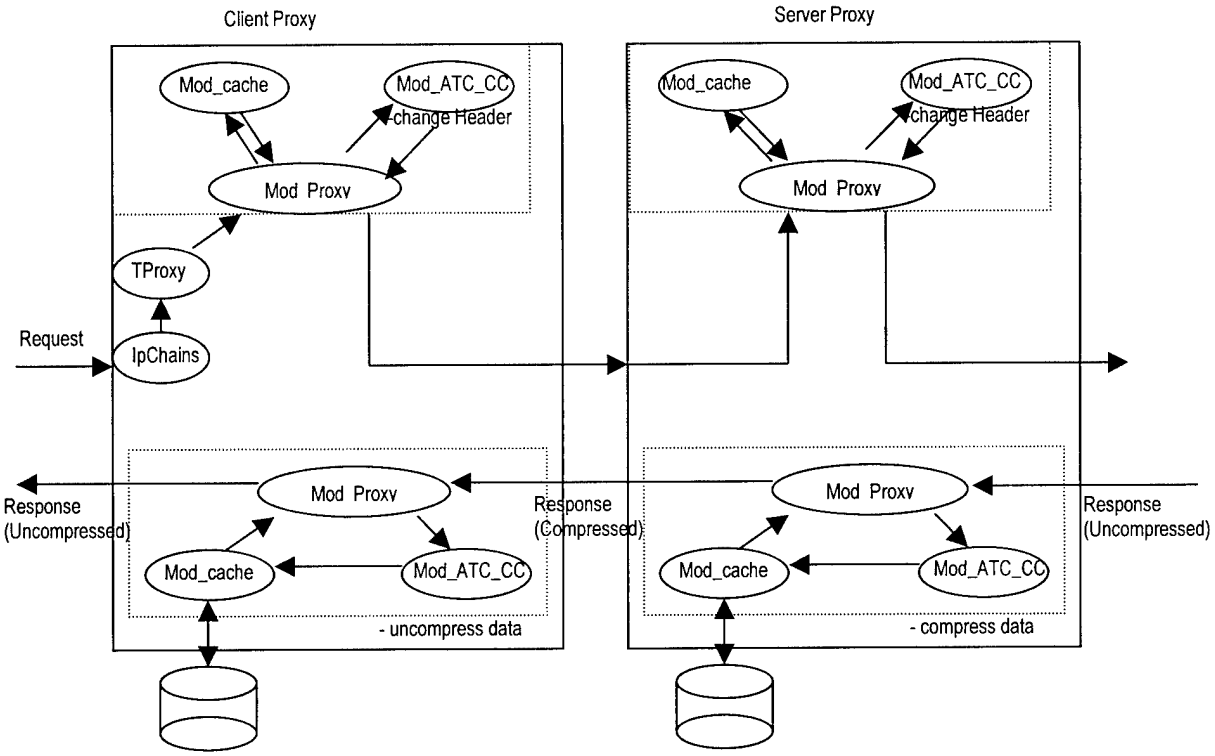
- It compresses the body (i.e. data) part of the RESPONSE packet,
- It caches the compressed version,
- It changes the header field of the RESPONSE packet, i.e. it sets the "content-encoding" values to "gzip, deflate, compress, etc.",
- It forwards the RESPONSE packet with the compressed data to the client proxy.

When the client proxy receives the RESPONSE packet:

- It decompresses the body (i.e. data) part of the RESPONSE packet,
- It caches the uncompressed version,
- It changes the header field of the RESPONSE packet, i.e. it sets the "content-encoding" values to null, so that the client is inhibited from invoking any decompression operations,
- It forwards the RESPONSE packet with the uncompressed data to the client.

These compression/decompression operations between proxies are totally transparent to the client and server sides. The figure below depicts the process flow of the pre-prototype proxy design when the file is stale in both caches.

[1] In the pre-prototype implementation, we are still, in effect, chunking, compressing the payload portion of each packet as it is received, but, unlike Transfer-Encoding practice, we are using varying compression methods. As a rule, greater efficiencies are obtained when compressing large objects than when compressing small objects. The fully developed Compression Controller will realize this greater efficiency potential by compressing complete files rather than chunks.



5.2 Cacheing Strategy [Not Yet Implemented]

Note that each proxy stores the file after having transformed it, i.e:

- The Server Proxy receives the file uncompressed from the Server, compresses it, stores it as compressed and sends the compressed file to the Client Proxy.
- The Client Proxy receives the compressed file from the Server Proxy, de-compresses it, stores it as uncompressed and forwards the uncompressed file to the Client.

In this way, each proxy is cacheing the file in the necessary format for transmission if it is still current on subsequent requests, thus avoiding compression or de-compression steps.

When the flow direction reverses (i.e., the HTTP pull is from the other end), the format in which the files are stored also reverses, thus the proxy caches will each contain both compressed and uncompressed files. Since the pulls from shipboard and those from shore will tend to differ, it is expected that most files will exist in the caches in a single format (either compressed or uncompressed). It is possible, however, that some will exist in both formats in the caches. The following scenarios illustrate the process savings provided by this approach:

- If the HTTP pull is from the shipboard client, and the file in the shipboard proxy cache is current, the uncompressed file will be retrieved from cache and sent to the shipboard client as is, avoiding a de-compression operation.
- In like manner, If the HTTP pull is from the shore client, and the file in the shore proxy cache is current, the uncompressed file will be retrieved from cache and sent to the shore client as is, avoiding a de-compression operation.

- If the HTTP pull is from the shipboard client, and the file in the shipboard proxy cache is not current, but the file in the shore proxy cache is current, the compressed file will be retrieved from the shore proxy cache and sent to the shipboard proxy as is, avoiding a compression operation.
- In like manner, If the HTTP pull is from the shore client, and the file in the shore proxy cache is not current, but the file in the shipboard proxy cache is current, the compressed file will be retrieved from the shipboard proxy cache and sent to the shore proxy as is, avoiding a compression operation.

5.3 Rationale

We have chosen the Proxy-to-Proxy approach for the following reasons:

- We need to be independent of the client and especially the server configurations:
  - Client configurations may change over time due to forces beyond the purview of this solution.
  - Server configurations may also change.
  - Servers not under control of the using command can be, and probably are, accessed.
- We wish to employ, selectively, compression techniques not in the current or projected repertoires of COTS products. It is important that we be able to import more advanced compression techniques as they become available and that we be able to apply those techniques which can most efficiently compress the type of file being compressed (e.g., image or text), both in terms of relative degree of compression and processor load required to perform the compression. It is also important to assess which mode of compression is most suitable to the QOS requirements of the file type or the user; i.e., should lossy or lossless compression be used?
- We wish to contain our solution, to the greatest degree possible, to augmentation, rather than modification, of existing fielded systems.

6 Experiments

6.1 Description

Limited experiments were performed in the Network Design Facility (NDF) at the Rome Research Site (RRS) of the Air Force Research Laboratory (AFRL). The purposes of these experiments were to demonstrate that the proxy was performing the functions it was designed to perform and to verify that there were no unexpected deleterious interactions with a wireless SATCOM environment. Although our major concern at this stage of development was that the Compression Controller not degrade performance, we were gratified to observe that it did in fact improve performance even in the absence of sophisticated rules based on more realistic traffic profiles and the matching of files to compression methods based on more informed analysis. These experiments were run *without* any other proposed enhancements to the system, such as SCPS-TP or MDP. The NDF, however, includes a component at the Link Layer (OSI Layer 2), the Configurable Protocol Stack (CPS), that performs one-hop retransmissions to alleviate problems with the wireless link, about which more later (see 6.3).

A directory was created on the Web server containing test files which could be downloaded individually and timed (stopwatch and *tcpdump*). The test files were chosen to roughly demonstrate both realistic Web access for Navy applications and to constitute a range of types and sizes that demonstrate varying compression potential and compression method applicability. The files used in the experiments were:

cctest.htm; HTML; 1kB; no compression; The home page for the tests. Links to the other files are listed for successive access and timing. This file resides on the test station disk and is not PULLed over the network or timed.

experiment.htm; HTML; 103kB; DEFLATE 9; Section 6 (this section) of an early version of this document, saved as HTML.

conventions.htm; HTML; 3kB; DEFLATE 1; A typical Web page explaining the site and how to use it. This file was copied from the site of the Fleet Numerical Meteorology and Oceanography Center (FNMOC). It provides a legend for interpreting weather maps supplied by the site and can be found at http://aipswebu1.fnmoc.navy.mil/MyWxmap/tutorials/Tutorial_Frame_Set.html.

weather.bmp; BMP; 58kB; COMPRESS; A weather map obtained from the FNMOC Website, showing wind speeds and directions in the Southwest Asia area. It was downloaded as a GIF file and saved as a BMP file to offer an opportunity for compression. It was also converted from color to B&W to reduce it to a size more practical for our testing schedule. This particular map, of course is no longer available, but similar maps can be viewed by starting at http://www.fnmoc.navy.mil/PUBLIC/.

recon.bmp; BMP; 41kB; COMPRESS; An advanced KH-11 photograph of the Shifa Pharmaceutical Plant, Sudan, downloaded in JPG format, cropped for size and converted to BMP format to offer an opportunity for compression. It was downloaded from http://www.gwu.edu/~nsarchiv/NSAEBB/NSAEBB13/6.jpg at The National Security Archive.

redball.gif; GIF; 326 bytes; no compression; A small "button" image of the sort common on Websites. It is included to provide a file which is not a candidate for compression in our test rule-set (see 7) in order to verify that compression was, indeed, not invoked.

Running with and without the Compression Controller active, each file was PULLed and timed via stopwatch. Traffic between the proxies was also monitored by *tcpdump* to record the sizes of the files being transferred over the proxy-to-proxy wireless link, for comparison of original sizes to compressed sizes, and to record packet by packet information for later analysis. The Compression Controller was instrumented to report its activity and to report the time used to compress and decompress. Time to compress ranged from .005 secs for *conventions.htm* to .1 secs for *weather.bmp*.

The experiment was performed over a serial link run at SATCOM data rates (2400bps, half duplex), and over an actual wireless SATCOM DAMA link using an AN/PRC 117-F radio. The tests over the degraded serial link were performed at least twice in order to verify typicality of the transmissions. Unfortunately, due to access time constraints, we were unable to obtain multiple readings over the DAMA link, the one most subject to variability. Fortunately, our single-pass tests with the Compression Controller active were all successful, thus confirming that the Compression Controller could operate over a satellite link. This, as has been stated earlier, was the major goal of the experiments.

Applications often timed out or failed in unpredictable ways, apparently due to the long and highly variable latency, confirming the need for link layer FEC (to reduce the frequency of ARQ retransmissions). The application failed less often with the Compression Controller than without, presumably due to compression having reduced the length of the transmitted files and thus the length of time required to transmit them -- the window of opportunity for something to go wrong.

Results are summarized in the tables below. Where there was more than one data point, a conservative approach was followed in selecting the one to use: for transfers not employing compression, the fastest transfer time was used; for transfers employing compression, the slowest transfer time was used.

6.2 Results

File Name	Original Size	Compressed Size	Compression Ratio	Time Without CC (seconds)	Time With CC (seconds)	Reduction in Time to Transfer
REDBALL.GIF	326	326	0	30	70	-133%
EXPERIMENT.HTM	103318	7253	93%	849	98	88%
CONVENTIONS.HTM	2821	1481	48%	51	48	6%
WEATHER.BMP	58922	30215	49%	499	272	45%
RECON.BMP	41482	37382	10%	364	342	6%

Table 1. Summary of Tests Over Serial Link Run At SATCOM Data Rates

File Name	Original Size	Compressed Size	Compression Ratio	Time Without CC (seconds)	Time With CC (seconds)	Reduction in Time to Transfer
REDBALL.GIF	326	326	0	N/A	75	
EXPERIMENT.HTM	103318	7253	93%	Transfer failed.	177	
CONVENTIONS.HTM	2821	1481	48%	N/A	76	
WEATHER.BMP	58922	30215	49%	785	397	49%
RECON.BMP	41482	37382	10%	N/A	560[2]	

Table 2. Summary of Tests Over Wireless SATCOM (DAMA)

6.3 Observed Effects

Certain effects were observed while running the experiments:

6.3.1 Layer 2 Retransmissions

During the SATCOM tests multiple retransmissions were observed at Layer 2. These were performed by CPS (see 6.1) in response to "bad CRC check" events (corruption in transit). CPS shielded TCP at Layer 4 from the experienced corruption and improved operations by retransmitting only 128 byte segments rather than the full 1480 byte packets that would have been retransmitted by TCP. We believe it probable that Forward Error Correction (FEC) at Layer 2 would have improved operations even more.

[2] The application timed out. This timing was obtained by observing the real-time tcpdump display and recording when the actual TCP transfer completed.

6.3.2 Layer 4 Retransmissions

At the beginning of each file download, multiple (two to eight observed) retransmissions of SYNs, SYN-ACKs and ACKs (the "three-way handshake") always occur, but there were no retransmissions of data. TCP learned the ambient latency of the circuit during the handshake phase (control) and applied it during the data transfer phase so no packets timed out and CPS at the link layer took care of corrupted packets. This behavior, unsurprisingly, occurred both with and without Compression Controller active. A transport protocol better suited to the wireless environment, or at least a standard transport protocol tuned for the environment, would, by eliminating this discovery phase, provide significant improvements in system responsiveness and reduction in bandwidth usage. For an estimation of potential improvements to be gained, see Appendix D.

6.3.3 Application Failures

Without Compression Controller, application failure was the norm. With Compression Controller, only one application failure was observed. Although we did not determine the cause of these failures, it may be due to time-out. If so, it should be easily fixed. We will investigate this during Phase II.

6.4 Analysis and Conclusions

The Compression Controller passed the primary test objective of these experiments by performing automatic, rule based, transparent compression of files before transmission over the wireless link and decompressing in like manner after transmission.

Except for the increased vulnerability of compressed files to high BER, it is a given that a compressed file, being smaller, will very likely be transmitted in less time than an uncompressed file, so improvement in transfer time, as observed in this experiment, should be no surprise. The results portrayed in the tables, however, merit some discussion and speculation as to their causes.

In the cable tests, the reduction in time to transfer *experiment.htm*, *weather.bmp* and *recon.bmp* was slightly less than the reduction in file sizes achieved by compressing them. This is seen as a result of our conservative choice of data points and the fixed overhead of TCP handshaking for setup and teardown of the connection.

The very large negative benefit shown in the data for *redball.gif* was puzzling. It may be that it was the result of factors in the system other than the Compression Controller since *redball.gif* was not compressed and constituted a single packet during both modes of operation (CC and no CC). We have been unable to rationalize this reason, however, and must consider the possibility of a glitch in the Compression Controller implementation. This will be investigated during Phase II.

conventions.htm exhibited little improvement in transmit time (6%) relative to its significant compression ratio (48%). The diminishing of returns achievable through compression as files become smaller is a well known effect. Another form of diminishing returns is illustrated in this example: the observed disappointing disparity between the compression ratio and the improvement in transmit time is due to the mechanics of the underlying system, in that a 2821 byte file, plus headers requires two TCP packets and a 1481 byte file, plus headers, will not fit into one packet and thus requires two packets, just as the larger version did. This illustrates that there is little or no benefit in compressing small files before transmission and is why we include file size as one of the criteria employed in deciding whether or not to compress. It might be useful in the ultimate Compression Controller to tune this algorithm by consulting the Maximum Transmission Unit (MTU) size when making the decision.

7 Compression Algorithm at Proxy

In its fully developed form, the Compression Controller will present an easy to use rule specification GUI to the System Administrator, or any other personnel allowed to change the compression rules.

Anticipated rule classes are:

- Type and size of file
- Acceptability of lossy compression artifacts
- Priority
- User ID

This part of the Compression Controller was not implemented in the prototype. The lines below constitute the simple, static rule set that was implemented and used in the experiments.

If (content-encoding == NULL) of the RESPONSE packet,
If (content-type == text/html)
if (content-length < 1KB)
Do not use any compression scheme.
Else if (content-length < 10KB)
Use ZLIB 'DEFLATE' compression scheme (Compression level = 1)
Else if (content-length < 20KB)
Use ZLIB 'DEFLATE' compression scheme (Compression level = 2)
Else if (content-length < 30KB)
Use ZLIB 'DEFLATE' compression scheme (Compression level = 3)
Else if (content-length < 40KB)
Use ZLIB 'DEFLATE' compression scheme (Compression level = 4)
Else if (content-length < 50KB)
Use ZLIB 'DEFLATE' compression scheme (Compression level = 5)
Else if (content-length < 60KB)
Use ZLIB 'DEFLATE' compression scheme (Compression level = 6)
Else if (content-length < 70KB)
Use ZLIB 'DEFLATE' compression scheme (Compression level = 7)
Else if (content-length < 80KB)
Use ZLIB 'DEFLATE' compression scheme (Compression level = 8)
Else
Use ZLIB 'DEFLATE' compression scheme (Compression level = 9)

Else if (content-type == image/bmp || image/png)
Use 'COMPRESS' compression scheme.
Else if (content-type == image/gif || mage/jpeg)
Do not use any compression scheme
Else
Do not use any compression scheme

Else If (content-encoding == deflate || compress) of the RESPONSE packet,
If (content-encoding == 'compress')
Use 'UNCOMPRESS' decompression scheme.
Else If (content-encoding == 'deflate')
Use ZLIB 'INFLATE' decompression scheme
Else
Do not use any decompression scheme

8 References

- Compressed Internet Protocol (IP) Data Via Geosynchronous Earth Orbit (GEO) Satellite Circuits Topic Number: N01-052; Phase I Proposal;

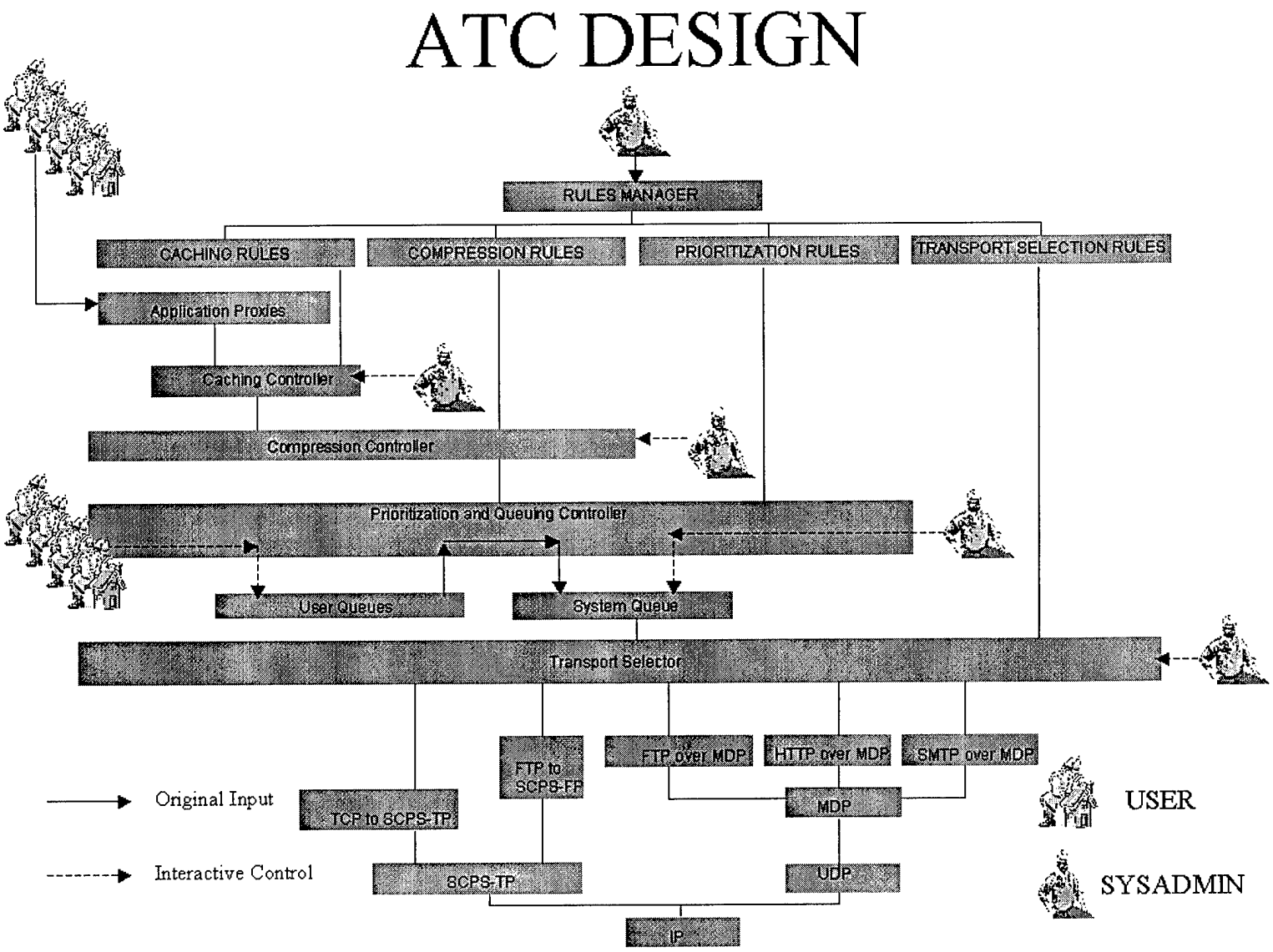
- Compressed Internet Protocol (IP) Data Via Geosynchronous Earth Orbit (GEO) Satellite Circuits Topic Number: N01-052; Phase I Technology Report Study (Final Report)

- Compressed Internet Protocol (IP) Data Via Geosynchronous Earth Orbit (GEO) Satellite Circuits Topic Number: N01-052; Phase II Proposal;

- Hypertext Transfer Protocol -- HTTP/1.1 (RFC2068); R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T.Berners-Lee; January 1997; ftp://ftp.isi.edu/in-notes/rfc2068.txt

Appendix A

Application Traffic Controller



Appendix B

Compression Controller

The Compression Controller segment of the Application Traffic Controller (ATC) will be responsible for controlling the implementation of data compression activity on both incoming and outgoing data at the Application Layer level. The intent is to provide the user with the benefits of compression techniques without requiring him to separately invoke the utilities needed to compress outgoing data or decompress incoming data. The Compression Controller will bear the responsibility of screening incoming and outgoing data and forwarding it on to an appropriate compression utility if necessary. This controller will have the ability to decide, based upon a set of predefined rules, whether or not the data requires use of a compression utility and which utility is most appropriate for the particular data. Navy system administrators will define these rules, based on the type of data being transmitted, the importance of the data (possibly based upon the priority specified), and possibly the source/destination of the data.

Invoking this controller at the Application Layer can exploit awareness of the format, content and intended use of the data. For example, standard images being transmitted in personal mail may be designated to use a lossy form of compression whereas images originating from the map room or a strategic planning area or even from a high ranking official may be directed to a lossless compression utility due to the importance of the data. Rules will need to be written carefully – for instance, not all JPEGs are created equal: weather maps, typically, may be compressed further relative to their source format, with little consequence; but target area imagery intended for use by intelligence analysts may not be able to tolerate additional compression artifacts.

In addition to improving ease of use of compression utilities and removing a possible distraction in the event that time critical tasks are being performed, the ATC will provide better control over the forms of compression being enacted in any given situation, resulting in reduced wireless WAN bandwidth requirements, better control over data flow, and better control over the quality of data as delivered.

Appendix C

HTTP/1.1 messages

The HTTP/1.1 protocol is based on request-response operations. A client sends a request to the server. The server responds with messages requested. HTTP/1.1 uses REQUEST and RESPONSE messages for the operations, respectively. The structure of each message is as follows.

REQUEST MESSAGE

Request-Line		OPTION|GET|HEAD|POST|PUT|DELETE|TRACE| URI H TTP1.1
Header	general-header	Cache-Control | Connection | Date | Pragma | Transfer-Encoding | Upgrade | Via
	Request-header	Accept | Accept-Charset | Accept-Encoding | Accept-Language Authorization | From | Host | If-Modified-Since | If-Match | f-None-Match | If-Range | If-Unmodified-Since | Max-Forwards | Proxy-Authorization | Range | Referer | User-Agent
	Entity-header	Allow | Content-Base | Content-Encoding | Content-Language | Content-Length | Content-Location | Content-MD5 | Content-Range | Content-Type | Etag | Expires | Last-Modified |
Message Body		It is determined by - Content-Type : specify media type - Content-Encoding : Compression

RESPONSE MESSAGE

Status-Line		H TTP1.1 Status-Code (1xx | 2xx | 3xx | 4xx | 5xx) Reason-Phrase
Header	general-header	Cache-Control | Connection | Date | Pragma | Transfer-Encoding | Upgrade | Via
	Response-header	Age | Location | Proxy-Authentication | Public | Retry –After | Server | Vary | Warning | WWW-Authenticate
	Entity-header	Allow | Content-Base | Content-Encoding | Content-Language | Content-Length | Content-Location | Content-MD5 | Content-Range | Content-Type | Etag | Expires | Last-Modified |
Message Body		It is determined by - Content-Type : specify media type - Content-Encoding : Compression

Appendix D

Effects of Re-Transmissions During the Handshake Phase of TCP

As has been mentioned earlier, there were no observed retransmissions of data packets at Layer 4. There were, however, many retransmissions during the handshaking phases. To get a rough idea of the effects of the retransmissions at Layer 4 during the setup and teardown phases when transmitting a file, we examined the *tcpdump* record of SATCOM tests with Compression Controller active. Table 1 illustrates the significant degradation caused by these retransmissions. "Goodput" in the table represents the percent of time taken by the transmission devoted to actual transfer of data. Disparities between these times and those derived from stopwatching the tests can be ascribed to the difference between when a transfer is complete at Layer 4 and when the result actually appears on the screen and the fact that these measurements were derived from the server-end *tcpdump* only.

File	Setup (A) (secs)	Data (B) (secs)	Teardown (C) (secs)	Goodput B/(A+B+C)
experiment.htm	59	66	60	.357
conventions.htm	20	24	31	.320
weather.bmp	18	219	176	.530
recon.bmp	40	305	197	.563

Table 1. Goodput Observed During Experiment

A transport protocol better suited to this environment, or perhaps even a standard one tuned for this environment, could avoid the needless retransmissions seen in this experiment. Table 2 presents a rough estimate of potential goodput. An average of four retransmits in the current case is assumed. Accordingly, setup and teardown times have been reduced by 80% for a case in which no Layer 4 retransmits occur.

File	Setup (A) (secs)	Data (B) (secs)	Teardown (C) (secs)	Goodput B/(A+B+C)
experiment.htm	12	66	12	.733
conventions.htm	4	24	6	.706
weather.bmp	4	219	35	.849
recon.bmp	8	305	39	.866

Table 2. Goodput With Appropriate Protocol

Applying this imputed improvement in goodput to the observed stopwatch timings, Table 3 indicates the potential improvement in delivery afforded by a more appropriate transport service. A corresponding reduction in bandwidth usage would also be achieved.

File	Stopwatch Time (secs)	Projected Time with Appropriate TCP*	% Improvement
experiment.htm	177	86	51%
conventions.htm	76	34	55%
weather.bmp	397	248	38%
recon.bmp	560	364	35%

Table 3. Potential Improvement with Appropriate Protocol

* Stopwatch (Observed Goodput / Imputed Goodput)